Protokol k semestrální práci z BI-ZUM

FIT ČVUT, LS 2019/2020

Jméno studenta: Peřina Jan
Username: perinja2
Název semestrální práce: SnAIk - AI solver pro hru snake

Zadání

Pomocí genetického algoritmu se pokuste vytrénovat váhy pro neuronovou síť, která bude hrát hru snake. Případně pomocí neuroevoluce vyšlechtěte takovou neuronovou síť.

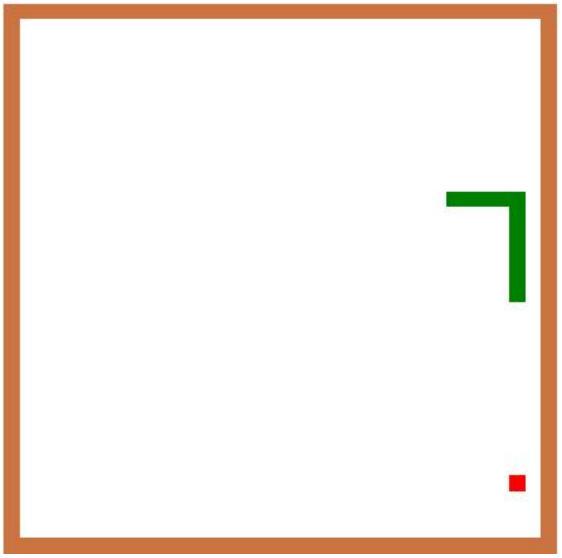
Rozbor

Cílem řešiče je maximalizovat počet jablek, které had ve hře sní, a tím tedy i dobu hraní. Aby se počet jablek zvyšoval, je za potřebí, aby řešič jednak naváděl hada k jablku a zároveň se vyhýbal zdem a hadu samotnému. Jelikož řešič vlastně odpovídá mozku hada, může pracovat pouze s informacemi, které má had k dispozici. V přírodě se had orientuje pomocí vidění a pachových vjemů, což můžeme napodobit prohledáváním mapy hry v zorném poli hada. Jaké informace jsou ale pro našeho hada stěžejní?

Implementace hry snake

K implementaci hry snake jsem se rozhodl použít 2d pole k reprezentaci mapy, jednotlivé objekty v poli obsahují informaci o typu objektu (had, jablko, zeď, volný prostror) a poskytují pomocné funkce, které usnadňují práci s hrou. Had jako takový je reprezentovaný listem, kde hlavě odpovídá objekt na poslední pozici listu a ocasu hada prvek první. Had se pohne, pokud je zavolaná metoda step. Počáteční pozice hada i jablka se vždy generují náhodně, avšak existuje vždy pouze jedno jablko. Had poté putuje mapou a pokud sní jablko, vyroste, pokud sní sebe, nebo narazí do zdi, umře. Jako metodu pro orientaci hada, jsem implementoval metodu sniff, která podle konfigurace vrátí vektor s hodnotami v závislosti na mapě. Pro grafickou reprezentaci vrací hra pole integerů, kde každý integer odpovídá typu objektu (had, jablko, zeď, prázdný prostor) a na základě kterého, je potom v jupyter notebooku aplikováno barevné schéma.

Hra jako taková je rozměrově škálovatelná a vypadá takto:



Aplikace metody

V první fázi jsem se rozhodl pro šlechtění vah plně propojené dopředné neuronové sítě za pomoci genetického algoritmu. Neuronová síť obsahovala 24 vstupních neuronů, dvě skryté vrstvy, každá o 64 neuronech a 4 výstupní neurony, které sloužily jako klasifikátory, k určení směru dalšího kroku hada. Vstupním vektorem do této neuronové sítě byl vektor o 24 neuronech, který obsahoval informaci o tom, v jaké vzdálenosti (L1-vzdálenost) je: zeď, had, jablko ve všech osmi směrech (bráno výskytově první dané kategorie), tedy 8*3 hodnot. V této fázi byla má implementace hry velice navní.

V této fázi jsem se pokoušel šlechtit váhy neuronové sítě, což v této konfiguraci vedlo k velké dimenzionalitě jedinců. V počáteční fázi šlechtění vah jsem použil genetický algoritmus s OnePoint operátorem křížení, Gausovský operátor mutace a Roullette operátor selekce. Jako hlavní algoritmus jsem použil algoritmus *Mu+lambda*, který v každé generaci vygeneruje *lambda* potomků a poté sjednotí aktuální generaci s potomky a následně z množiny vybere *lambda* jedinců do nové generace.

Gausovský operátor mutace

Gausovský operátor mutace funguje tak, že s pravděpodobností *p* ke každé složce jedince přičte hodnotu z gausovského rozdělení z parametrů (*mu*, *sigma).*

Tato konfigurace však v řešení nebyla velice úspěšná. I po 100 generacích s počáteční generací 10000 jedinců a několika pokusech jsem nedosáhl viditelného výsledku, had buď ignoroval jablko a chodil po mapě, nebo opakoval stejný pattern pohybu (například dolu -> zahnout před zdí vlevo -> rovně -> narazit do zdi).

Změnil jsem proto jak architekturu neuronové síťě, abych zmenšíl prohledávaný prostor řešení, složky jedince jsem v počáteční generaci omezil na hodnoty z intervalu <-1, 1>, včetně změny parametrů *mu* a *lambda* v operátoru mutace a předělal jsem také způsob, jakým funguje funkce *sniff* včetně funkcionality hry.

Od teď mohl had provést pouze tři akce: pokračovat ve směru, zahnout vlevo, zahnout vpravo. Nová neuronová síť už nepracovala se vstupy jako s reálnými parametry, nýbrž jako existenčními. Nově jsem registroval ve směru vlevo, vpřed a vpravo, jestli je možné vstoupit na první, popř druhé políčko ve směru a jestli se jablko v tomto směru nachází. Dále jsem přidal cosinovou vzdálenost mezi jablkem a hlavou hada. Dále jsem přidal informaci o relativní pozici jablka vůči hadovi na horizontální a vertikální ose.

Napříč všem novým změnám a různou konfigurací operátorů, se had dostal na průměrné skóre 3-4 jablek (průměrná doba - 6h.). Což nebylo velmi uspokojující. Důvodem, proč toto řešení nebylo úspěšné, je podle mě fakt, že všechny příznaky byly stejně důležité, což zvětšovalo komplexnost hledání řešení. Had se vesměs dobře naučil vyhýbat zdem, avšak jablka konzumoval náhodně, nebo jsem odpozoroval vzorek, kde had snědl jablko, pouze pokud k němu šel ze spodu a měl ho na levé straně.

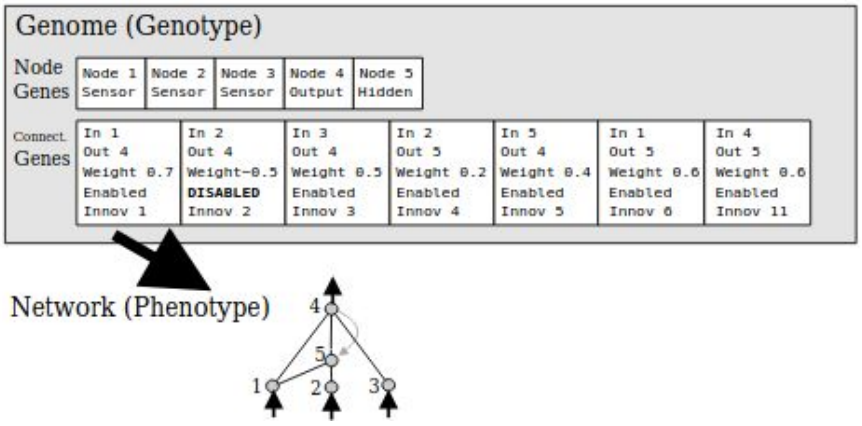
I přes nedobré výsledky jsem se rozhodl, že to zvládnu lépe. A proto jsem přešel na metodu NEAT (neuro-evolution through augmenting topologies), které na rozdíl od pouhého šlechtění vah, šlechtí také topologii neuronových sítí, společně s aktivačními funkcemi, atp..

NEAT

Algoritmus NEAT řeší pět hlavních problémů, které se naskytnou při šlechtění neuronové sítě. Jmenovitě se jedná o:

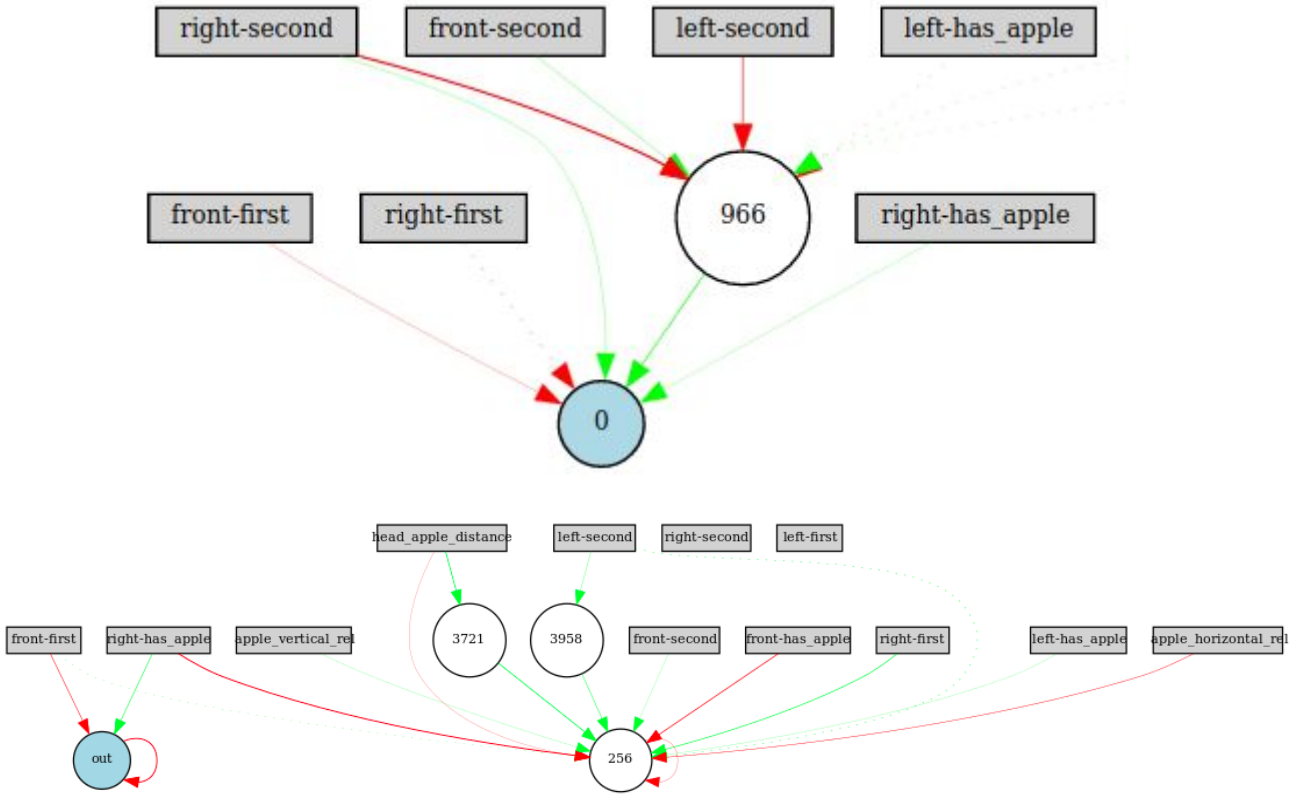
- Genetické zakódování
  Kódování topologie, vah, atp..
- Křížení neuronových sítí
- Rozdělení do druhů
  Jedná se o rozdělení jedinců do skupin - druhů (species), v rámci kterých soutěží a jsou penalizováni. Nedochází tedy k porovnávání fitness mezi všemi jedinci, nýbrž v menších skupinkách.
- Sdílení fitness (napříč druhy)
- Minimální počáteční struktura
  Neuronové sítě v první fázi jsou malé a postupně dochází k jejich kombinování, je-li to třeba.

S touto metodou jsem při prvním pokusu dosáhl za pouhých 6 generací řešiče, který byl schopný nasbírat 26 jablek. Poté už bylo potřeba pouze odladit konfiguraci NEAT, podle které docházelo k šlechtění neuronových sítí.



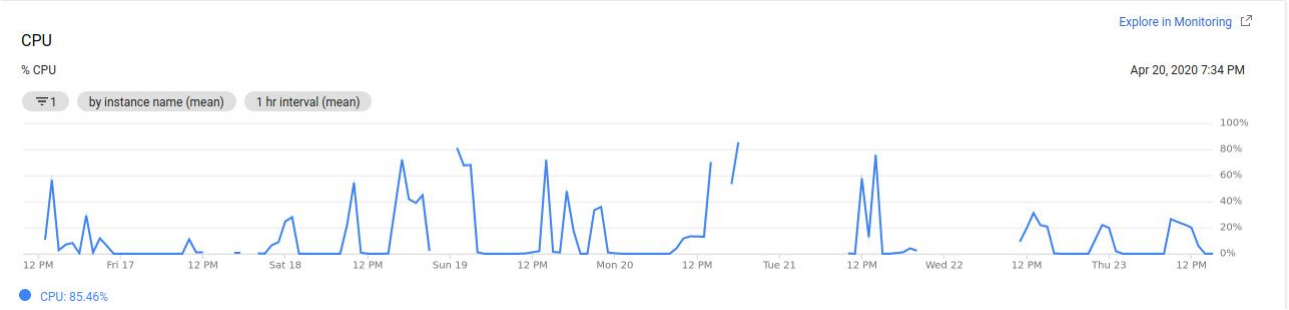
Převod genotypu na fenotyp v NEAT

Příklady vyšlechtěných síťí



Závěr

Jak jsem se dočetl z NEAT paperu, šlechtění vah neuronové sítě s fixní topologií je násobně složitější a více náchylné na chyby. Zvláště pokud neuronová síť není primitivní, může dojít k několika hodinovému trénování s mizivým výsledkem. NEAT naopak poskytuje sofistikovaný proces šlechtění a nalezení řešení bylo v mém případě rychlejší.



Vytížení procesoru při šlechtění vah

Reference

Evolving Neural Networks through Augmenting Topologies - Stanley, Miikkulainen